(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 261 139 A2

(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
27.11.2002 Bulletin 2002/48

(51) Int Cl.7: **H03M 13/29**, H03M 13/41

(21) Application number: **02100509.5**

(22) Date of filing: **20.05.2002**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **23.05.2001 US 293014 P**

(71) Applicant: **Texas Instruments Incorporated**
Dallas, Texas 75251 (US)

(72) Inventor: **WOLF, Tod D.**
Richardson, TX 75801 (US)

(74) Representative: **Holt, Michael**
Texas Instruments Ltd.,
EPD MS/13,
800 Pavilion Drive
Northampton Business Park,
Northampton NN4 7YL (GB)

(54) **Concurrent memory control for turbo decoders**

(57) The concurrent memory control turbo decoder solution of this invention uses an interleaved forward-reverse addressing with a single port memory and a simplified scratch memory. This invention computes beta state metrics for a sliding window and stores them in a scratch memory in a first addressing order (1000). For each sliding window, this invention computes alpha state metrics for the sliding window, reads beta state metrics from the scratch memory in the addressing order of the current repetition (1005) and combines alpha state metrics and beta state metrics in an extrinsic block. This invention then computes beta state metrics for a next sliding window and stores the data scratch memory in the addressing order of the current repetition. The addressing order is toggled.(1001) for the next repetition (1004) until a frame of data is decoded.

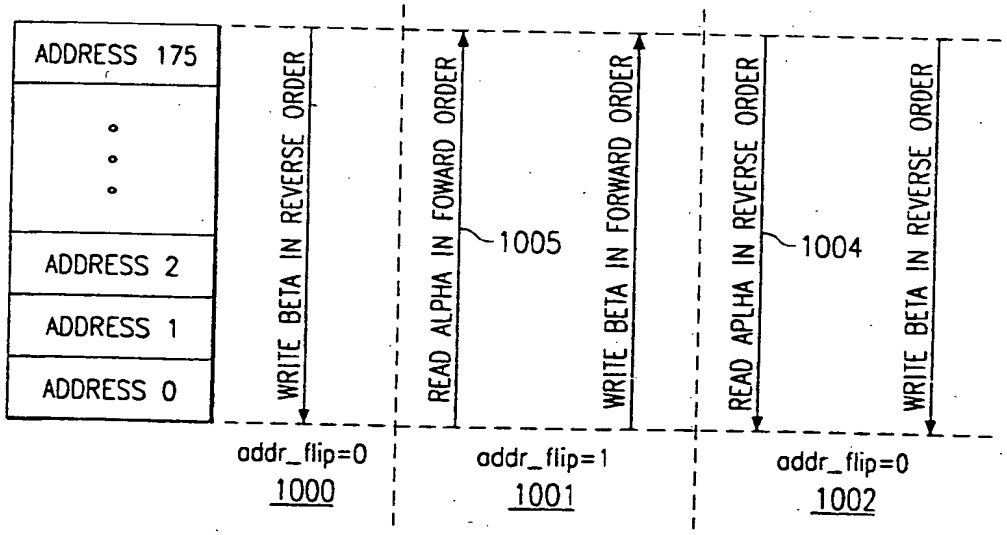

FIG. 10

EP 1 261 139 A2

Description

TECHNICAL FIELD OF THE INVENTION

[0001] The technical field of this invention is turbo decoders used in forward error correction.

BACKGROUND OF THE INVENTION

[0002] Turbo codes are a type of forward error correction code with powerful capabilities. These codes are becoming widely used in many applications such as wireless handsets, wireless base stations, hard disk drives, wireless LANs, satellites, and digital television. Turbo codes consist of a concatenation of convolutional codes, connected by an interleaver, with an iterative decoding algorithm.

[0003] An example of a prior art rate 1/3 parallel-concatenated encoder is shown in Figure 1.

[0004] Input data stream 100 (x_m) is supplied unmodified to multiplexer 104 at input 106. The two Recursive Systematic Convolutional (RSC) encoders 102 and 103 function in parallel to transform their respective input bit streams. After transformation by RSC encoders 102 and 103, the resulting bit streams are supplied to multiplexer 104 at inputs 107 and 108, respectively. Block 101 is an interleaver (I) which randomly re-arranges the information bits to decorrelate the noise for the decoder. RSC encoders 102 and 103 generate respective $p0_m$ and $p1_m$ bit streams. Multiplexer 104 reassembles these x_m, $p0_m$ and $p1_m$ bit streams into a resulting output bit stream 105 (x_0, $p0_0$ and $p1_0$ ...).

[0005] Figure 2 illustrates a functional block diagram of a prior art turbo decoder 200. Iterative turbo decoder 200 generates soft decisions from a pair of maximum-a-posteriori (MAP) blocks 202 and 203. Each iteration requires the execution of two MAP decodes to generate two sets of extrinsic information. The first MAP decoder 202 uses the non-interleaved data as its input and the second MAP decoder 203 uses the interleaved data from the interleaver block 201 as its input. The MAP decoders 202 and 203 compute the extrinsic information as:

$$W_n = \log \frac{Pr(x_n = 1 | R_1^n)}{Pr(x_n = 0 | R_1^n)} \quad [1]$$

where: $R_1^n = (R_0, R_1, \ldots R_n)$, which are the received symbols. MAP decoders 202 and 203 also compute the a posteriori probabilities:

$$Pr(x_n = 1 | R_1^n) = \frac{1}{Pr(R_1^n)} \Sigma\, Pr(x_n=i, S_n=m', S_{n-1}=m) \quad [2]$$

where: S_n is the state at time n in the trellis of the constituent convolutional code.

[0006] The terms in the summation can be expressed in the form

$$Pr(X_n=i, S_n=m', S_{n-1}=m) = \alpha_{n-1}(m)\, \gamma_n^i (m, m')\beta_n(m') \quad [3]$$

where: the quantity

$$\gamma_n^i (m,m') = Pr(S_n=m', x_n=i, R_n | S_{n-1}=m) \quad [4]$$

is called the branch metric, the quantity

$$\alpha_n(m') = Pr(S_n=m', R_1^n) \quad [5]$$

is called the forward (or alpha) state metric, and the quantity

$$\beta_n(m') = Pr(R_{n+1}^n | S_n=m') \quad [6]$$

is called the backward (or beta) state metric.

[0007] The branch metric depends upon the systematic, parity, and extrinsic symbols. The extrinsic symbols for each MAP decoder are supplied to the other MAP decoder at inputs 209 and 210. The alpha and beta state metrics are computed recursively by forward and backward recursions given by:

$$\alpha_r(m') = \sum_{m',i} \alpha_{n-1}(m)\gamma_n^i(m,m') \qquad [7]$$

and

$$\beta_{n-1}(m) = \sum_{m',i} \beta_r(m')\gamma_n^i(m,m') \qquad [8]$$

[0008] The slicer 207 completes the re-assembling of the output bit stream 208 ($x_0, x_1 \ldots x_{n-1}$).

[0009] Figure 3 illustrates a block diagram of a prior art MAP decoder. The subscripts r and f represent the direction, reverse and forward, respectively, of the sequence of the data inputs for the recursive blocks beta and alpha. Input bit streams 310 to 312 are labeled as parameters $X_{n,r}$, $P_{n,r}$ and $A_{n,r}$, respectively. Input bit streams 313 to 315 are labeled as parameters $X_{n,f}$, $P_{n,f}$ and $A_{n,f}$, respectively. The feedback stream from alpha state metric block 302 is labeled $\alpha_{n,f}$. The feedback stream from beta state metric block 303 is labeled $\beta_{n,r}$. Both the alpha state metric block 302 and beta state metric block 303 calculate state metrics. Both start at a known location in the trellis, the zero state. The encoder starts the block of n information bits (for example, n = 5114, the frame size) at the zero state and after n cycles through the trellis ends at some unknown state.

[0010] Without sliding windows, the frame size of the block would contain n x s x d = 327,296 bits. With sliding windows, the processing involves r x s x d = 8192 bits where r is 128. Clearly, the memory size requirements are greatly reduced through the use of sliding windows.

[0011] A number of tail bits t are appended to the encoder data stream to force the encoder back to the zero state. For a constraint length k code, t=k-1, there are systematic tail bits for each RSC encoder. For an eight state code, k=4, t=3 which is assumed for the remainder of this description. Alpha state metric block 302 will process the data from 0 to n+2 and beta state metric block 303 will process the data from to n+2 to 0.

[0012] The beta state metrics are generated first by beta state metric block 303. These beta metrics are generated in reverse order and stored in the beta state metric RAM 304. Next, the alpha state metrics are generated by alpha state metrics block 303. The alpha state metrics are not stored because extrinsic block 305 uses this data as soon as it is generated.

[0013] The beta state metrics are read in a forward order at the same time as the alpha state metrics are generated. Extrinsic block 305 uses both the alpha and beta state metrics in a forward order to generate the extrinsic outputs 306 $W_{n,i}$. This implementation requires a large main memory RAM supplying the a-priori inputs 310 to 315. The main memory size is computed as listed in Table 1.

Table 1

Main Memory Size	Number of Bits
X_0	5120×8=40,960
P_0	5120×8=40,960
P_1	5120×8=40,960
A_0	5120×8=40,960
A_1	5120×8=40,960
I	5120×13=66,560
S_X	176×45×4=31,680
P_2	2560×8=20,480
P_3	2560×8=20,480

Table 1 (continued)

Main Memory Size	Number of Bits
Totals	344,000 bits

[0014] The size of the beta state metric memory can also be reduced by using the sliding block implementation. The block of size n is broken into smaller pieces of size r shown in Figure 4. Each smaller block of size r, called the reliability size, can be processed independently of each other by adding a prolog section of size p to each block of r.

[0015] The sliding window block is shown in Figure 5. The reliability size 501 is r. The prolog size 502 is p and is usually equal to 4 times to 6 times the constraint length. Upon setting all the state metrics to a zero and then executing the prolog, the resulting state metric has a high probability of being in the correct state. This block has a size of r+p. The size of the beta state metric memory will drop to r x 8 x d. Note that the state metrics for the beta prolog section are not stored.

[0016] The turbo decoder controller is required to process an entire frame of data. If the frame is large, then it must be processed into k smaller pieces or sub-blocks as shown in Figure 6. Each sub-block such as 600 or 601 consists of four sliding windows in this example. Of course, other groupings of sliding windows could have been used.

[0017] The beta sub-block must be processed and stored before the alpha and extrinsic (labeled extr in Figure 6) sub-blocks can start. Therefore, it takes some amount of time units to process k sub-blocks. Each sub-block consists of four sliding windows that are shown in Figure 7 and Figure 8. The arrows represent the processing order. RBx is the abbreviation for the reliability section for beta and PBx is the abbreviation for the prolog section for beta. Figure 8 illustrates the corresponding labels for alpha metrics.

[0018] When both beta and alpha sub-blocks are being processed simultaneously, the data memories must be accessed twice. Unfortunately, the addresses are different thus requiring a dual port memory. Other solutions are possible using a single port main memory combined with a combination of scratch memory blocks. Such implementations are hampered by the complexity involved in meeting the required addressing order. The scratch memory would include four separate memory blocks, one for each sliding window. Each of the scratch memory blocks would have 176 addressable locations; the sum of the maximum sizes for reliability and prolog. Each one of the four scratch memory blocks would store the data for one of the four alpha sliding windows.

[0019] The difficulty with this solution is that the beta data is written to the scratch memories in a reverse order and the alpha data is read in a forward order. This would require two memories for each sliding window to insure that the data is being processed correctly. During processing of the first sub-block, one of the memories is performing a write. During processing of the second sub-block, the full memory is read from for alpha processing and the other memory is written to for future alpha processing. During the processing of the next sub-block, the operation of the memories is reversed. This technique is call ping-ponging of memories. The memories are ping-ponged until each sub-block has been processed.

[0020] A conventional turbo decoder using the dual port main memory approach is illustrated in Figure 9. Blocks of data to be decoded 900 come from the digital signal processor (DSP) to the main memory 902. Main memory 902 is a dual-port RAM. Memory control block 901 generates both addresses 911 for main memory 902 and addresses 906 the beta RAM 907. Data is passed to the alpha metrics block 904 and the beta metrics block 905 from two separate ports of main memory 902. Beta metrics block 905 writes its output to beta RAM 907 and the alpha metrics block 904 passes its output directly to the extrinsic block 909. Because the output of beta metrics block 905 is used in the order described in Figures 6 and 7, a ping-pong beta RAM of a full 8-block size must be used. The multiplexer 908 provides interface between the eight separate portions of beta memory 907 and extrinsic block 909. Extrinsic block 909 completes computation of metric output parameters 910 W_{nj}.

[0021] To avoid loss of processor cycles, the conventional turbo decoder system of Figure 9 requires a dual port main memory 902 having an array size almost double the size of a single port memory. It also requires an eight-block beta memory 907 because of the order in which beta metrics output is used in comparison to the order in which the alpha metrics output is used in computing output extrinsic data 910 W_{nj}.

SUMMARY OF THE INVENTION

[0022] This invention is a concurrent memory control solution for turbo decoders. A four-sliding windows preferred embodiment requires only a single port main memory, a scratch memory and a four-block beta memory. This is in contrast to conventional turbo decoders which would employ a dual port main memory and an eight block size ping-pong beta memory. During each cycle, one read and one write operation must happen for the scratch memories. If a particular location in memory, has been read, then that location is free. The next write cycle can use that location to store its data.

[0023] During processing of the first beta sub-block the data memories for the systematic, parities, and a-priori are read. The reliability portion of this data is written into the scratch memory in a reverse order. After the beta sub-block processing has finished, the alpha reliability data is loaded into the scratch RAM, but not the alpha prolog data. The turbo decoder controller starts a new state in which the alpha prolog data is read from the data memories and the data is stored in the scratch RAM. The maximum size of each of the scratch memories is equal to the maximum sum of the reliability and prolog sizes. A solution for the addressing requirements for interleaved forward and reverse addressing order are also described.

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] These and other aspects of this invention are illustrated in the drawings, in which:

Figure 1 illustrates the high level functional block diagram of a prior art turbo decoder;

Figure 2 illustrates a lower level functional block diagram of a prior art turbo decoder;

Figure 3 illustrates a functional block diagram of a prior art MAP decoder;

Figure 4 illustrates breaking a block of size n into sliding window blocks of size r according to the prior art;

Figure 5 illustrates the make-up of a prior art beta sliding block;

Figure 6 illustrates the prior art processing of beta and alpha sub-blocks versus time;

Figure 7 illustrates the prior art processing of four beta sliding windows in parallel;

Figure 8 illustrates the prior art processing of four alpha sliding windows in parallel;

Figure 9 illustrates the prior art use of ping-pong scratch memory in a four-sliding-windows conventional turbo decoder;

Figure 10 illustrates the physical address order of scratch RAM in a first embodiment of this invention;

Figure 11 illustrates the processing of four beta sliding windows in parallel for a second embodiment of this invention;

Figure 12 illustrates the physical address order of scratch RAM for the second embodiment of this invention;

Figure 13 illustrates the processing of beta and alpha sliding windows versus time; and

Figure 14 illustrates the concurrent memory control of this invention with interfacing for main memory, scratch memories and beta memory in a four-sliding-windows turbo decoder.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0025] With four sliding windows assumed, Figure 10 illustrates the physical address order of scratch RAM for the preferred embodiment. Beta processing state takes a maximum of

$$((128+48)x4)+12 = 716 \text{ cycles} \quad [9]$$

and the alpha prolog processing state takes

$$(48x4)+8 = 200 \text{ cycles.} \quad [10]$$

[0026] The 12 term in equation 9 and the 8 term in equation 10 arise from the extra cycles needed to setup the respective states.

[0027] The 4 factor in equation 9 represents the number of sliding windows in this implementation. The main data memories are read (128+48)x4) = 704 times and the scratch memories are written (128x4) = 512 times during the beta processing state. The beta prolog data is not stored in the scratch memories during the beta processing state.

[0028] The problem of the write and read order is solved with the addition of virtual addresses. The virtual addresses for the beta are always indexed in the reverse order and the virtual addresses for the alpha are always indexed in the forward order. But, the physical addresses are mapped depending on signal addr_flip. Referring to Figure 10, the binary signals addr_flip 1000, addr_flip 1001, addr_flip 1002 toggle for each sub-block which causes the write and read order to change. When addr_flip is a 0, the physical addresses for both beta writes and alpha reads accesses the memory in reverse order 1004. When the addr_flip state is a logic 1, the physical addresses will access the memory in forward order 1005. This addressing scheme allows the scratch memory to be implemented with only one memory. There is one scratch memory for each of the four sliding blocks. This technique is also used for the beta state metric memory.

[0029] The disadvantage of the above addressing scheme is that it takes cycles. The MAP decoder is not being used during the 200 cycles it takes to store the alpha prolog data. This is a waste of 21.8% of the cycles.

[0030] Referring to Figure 11, the alpha prolog data for sliding window 1 (PA1) 1100 is part of the beta reliability data for sliding window 0 (RB0) 1101. Storing this data to the scratch memories during the beta processing state eliminates this function in the alpha prolog state. This eliminates 716+200 = 916 cycles from the alpha prolog state. This technique obviates the need for only three of the four alpha prolog sections because the data for the first alpha prolog sliding window (PA0) 1102 is not available during the beta processing state.

[0031] Further, the physical addressing in Figure 10 will not function properly with this solution. Writing the prolog alpha sections early in the beta processing state overwrites the previously stored reliability data from the last sub-block. This overwriting cannot be allowed because it causes the MAP decoder to function improperly.

[0032] Figure 12 shows a solution which avoids this difficulty. The scratch memory is divided into two regions. One region 1200 is for the reliability data and the other region 1201 is for the prolog data. The reliability region 1200 is controlled in a similar fashion as described above. The only difference is that the reliability address pointer is not allowed to go into the prolog address region 1201. The reliability address is still controlled by the addr_flip signals 1202, 1203 and 1204, as shown in Figure 12.

[0033] A second address pointer for prolog address region 1201 is added. The reading and writing of data to the prolog address region 1201 is simpler than to the reliability address region 1200. The reading and writing of prolog data with respect to time never overlap with each other as shown in Figure 13. Both the alpha and beta prolog data are read and processed at the beginning of the state 1300. The beta prolog data is not stored in the alpha scratch RAMs. Once the alpha prolog section has finished executing, then the prolog section is free in the scratch memory. When the beta starts the beta reliability section during time frame 1301, this data is stored twice, once in the current sliding window reliability address region 1200 and then in the next sliding window prolog address region 1201. The first part of the beta reliability data is the alpha prolog address region 1201 for the next sub-block as shown in Figure 11.

[0034] This requires a new memory addressing technique with two writes and one read operation every cycle during the critical part of the reliability section. There are four scratch memories, one for each of the alpha sliding windows. Offsetting the three accesses to a single memory by one cycle allows this system to perform properly. Therefore, three out of the four scratch memories are accessed during every cycle. Each one is accessed only once, therefore, allowing the physical implementation of the memory to be simple.

[0035] This new technique requires 716+56 = 772 cycles. This is a savings of 144 cycles over the number of cycles required in the first embodiment of the invention. These 144 cycles become significant when summing the number of cycles it takes to complete a turbo decode. For example, if the frame length is 5114 and 10 iterations are performed, each turbo decode requires the following number of cycles shown in Table 2.

[0036] The second embodiment of this invention requires fewer cycles compared to the first embodiment. That is a 13.3% cycle improvement.

Table 2

State	Equation for Second Embodiment	Number of Cycles for First Embodiment	Number of Cycles for Second Embodiment
Determine sliding windows pointers	(10+1)×4	44	44
beta,alpha,extrinsic processing	(10+1)((128+48)×4+12)	7876	7876
load alpha prolog data for sliding window '0'	(9)((48×1)+8)+(2×3)		510

Table 2 (continued)

State	Equation for Second Embodiment	Number of Cycles for First Embodiment	Number of Cycles for Second Embodiment
4 Sliding Windows	(9)((48×4)+8)+(2×3)	1806	
wait for extrinsics	(10×1)+(1×2)	12	12
start new sub-block	11×1	11	11
wait for stopping criteria	10	10	10
Total per MAP decode	m	9,759	8,463
Total per Iteration	$i = 2 \times m$	19,518	16,926
Total per 10 iterations	$10 \times i$	195,180	169,260

[0037] Figure 14 illustrates a block diagram of a MAP decoder architecture using the concurrent memory control of a preferred embodiment of this invention. This preferred embodiment is a four-sliding-windows architecture which requires four scratch memories and four beta memories. Four sliding window data is efficiently processed in a four-cycle beta metrics block architecture. Figure 14 is an expanded view of Figure 3, showing blocks of data to be decoded 1400 coming from the digital signal processor (DSP) to main memory 1402. Concurrent memory controller 1401 provides addresses 411 for main memory 1402, addresses 1412 for scratch memory 1403 and addresses 1406 for beta RAM 1407. Alpha metrics block 1404 and beta metrics block 1405 both interface with the scratch memory 1403. Beta metrics block 1405 writes to scratch memory 1403 and alpha metrics block 1404 reads from scratch memory 1403. Concurrent memory interface controller 1401 controls all memory operations in main memory 1402, scratch memory 1403 and controls beta RAM 1407. Scratch memory 1403 employs 45 bit scratch memory words consisting of systematic bits (8), parity bits (8X2=16), a-priori bits (8) and interleaver data (13). The interleaver data is the extrinsic data address used when storing the extrinsic information. Concurrent memory controller 1401 drives the flow of data according to the prescription of Figures 10 through 12. It performs control and address generation for all three memory blocks. Multiplexer 1408 provides interface between the four separate portions of beta memory 1407 and the extrinsic block 1409. Extrinsic block 1409 completes computation of the metric output parameters 1410 W_{nj}.

[0038] Turbo coders are becoming widely used in many fields of communications. Turbo decoders are iterative decoders which execute the MAP decoder twice per iteration. The typical number of iterations ranges from 6 to 12. It is important to reduce the cycle count per decode which improves the system performance. A novel approach to limiting the number of memories required and a method of controlling the memories efficiently is described here. Most of the alpha prolog data and all of the alpha reliability data is folded into the cycles required to generate the beta state metrics. This reduces the cycle count of the decode.

Claims

1. A method of turbo decoding comprising the steps of:

   computing beta state metrics for a sliding window of data,
   storing said beta state metrics for the sliding window of data in a scratch memory in a first addressing order;
   then repetitively for each sliding window
   toggling a current addressing order for a current repetition between the first addressing order and a second addressing order, said second addressing order being opposite to said first addressing order,
   computing alpha state metrics for the sliding window of data,
   reading beta state metrics from the scratch memory in the addressing order for the current repetition,
   combining the computed alpha state metrics and the recalled beta state metrics in an extrinsic block thereby producing extrinsic outputs,
   computing beta state metrics for a next sliding window of data,
   storing said beta state metrics for the next sliding window of data in a scratch memory in the addressing order of the current repetition;

   until a frame of data including a plurality of sliding windows is decoded.

2. The method of turbo decoding of claim 1, wherein:

the step of toggling an addressing order for a current repetition includes toggling a digital state of an address flip bit controlling an order of mapping of a virtual address to a physical address, wherein storing said beta state metrics always occurs in a third addressing order in virtual memory addressing and reading said beta state metrics always occurs in a fourth addressing order in virtual memory addressing, the fourth addressing order opposite to said third addressing order.

3. A method of turbo decoding comprising the steps of:

providing a plurality of scratch memories, each of said scratch memories having a reliability portion and a prolog portion, the plurality of scratch memories employed in a predetermined circular order;
computing beta state metrics for an initial sliding window of data,
simultaneously storing said beta state metrics for the sliding window of data in the reliability portion of a current scratch memory in a first addressing order and in the prolog portion of a next succeeding scratch memory in a second addressing order;

then repetitively for each sliding window
toggling a current addressing order for a current repetition between the first addressing order and a third addressing order, said third addressing order being opposite to said first addressing order,
computing alpha state metrics for a current sliding window of data,
reading beta state metrics from the prolog portion of the current scratch memory in a fourth addressing order, the fourth addressing order opposite to the second addressing order,
reading beta state metrics from the reliability portion of the current scratch memory in the current addressing order,
combining the computed alpha state metrics and the recalled beta state metrics in an extrinsic block thereby producing extrinsic outputs,
computing beta state metrics for a next sliding window of data,
simultaneously storing said beta state metrics for the next sliding window of data in the reliability portion of a next succeeding scratch memory in an addressing order opposite to the current addressing order and in the prolog portion of a second succeeding scratch memory in the second addressing order;
until a frame of data including a plurality of sliding windows is decoded.

4. The method of turbo decoding of claim 1, wherein:

the step of toggling an addressing order for a current repetition includes toggling a digital state of an address flip bit controlling an order of mapping of a virtual address to a physical address, wherein storing said beta state metrics in a reliability portion of a scratch memory always occurs in a fifth addressing order in virtual memory addressing and reading said beta state metrics from the reliability portion of a scratch memory always occurs in a sixth addressing order in virtual memory addressing, the sixth addressing order opposite to said fifth addressing order.

5. A turbo decode apparatus comprising:

a main memory storing data to be turbo decoded;
a beta metrics block connected to said main memory for forming beta metrics from data recalled from said main memory;
a set of a plurality of scratch memories connected to said beta metrics block for storing said beta metrics, each scratch memory having a prolog section and a reliability section;
an alpha metrics block connected to said set of scratch memories for forming alpha metrics from data recalled from the scratch memories;
a beta metrics memory connected to said beta metrics block storing beta metrics formed by said beta metrics block, said beta metrics memory having a plurality of sections;
a multiplexer having a plurality of inputs each connected to a corresponding one said beta metrics memory and an output outputting data recalled from a selected one of said sections of said beta metrics memory;
an extrinsic block connected to said alpha metrics block and said multiplexer for forming extrinsic outputs; and
a concurrent memory interface controller connected to said main memory, said set of a plurality of scratch memories and said beta metrics memory, said concurrent memory interface controller generating addresses whereby said beta metrics block writes to said prolog section of a first scratch memory in a first addressing order, said beta metrics block writes to said reliability section of a next following scratch memory is a second

alternating addressing order, said alpha metrics reads from a second following scratch memory in a third addressing order opposite to said first addressing order, and said alpha metrics reads from a third following scratch memory in a fourth alternating order opposite to said second alternating order.

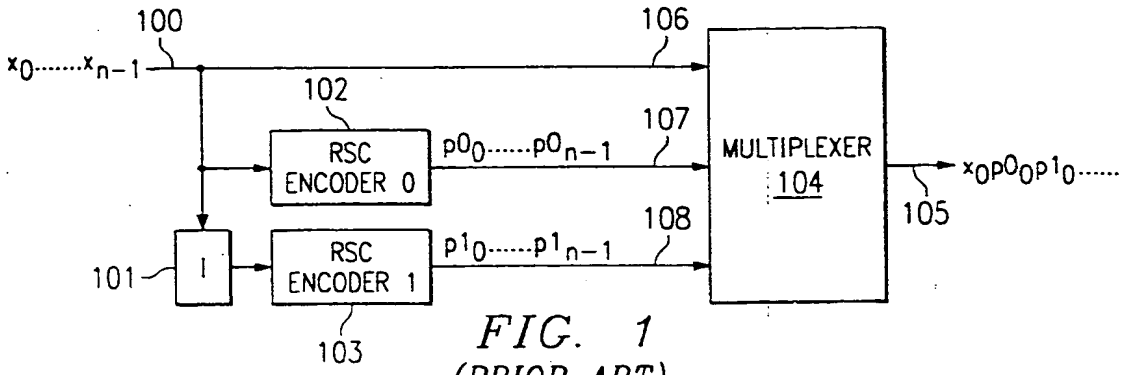

FIG. 1
(PRIOR ART)

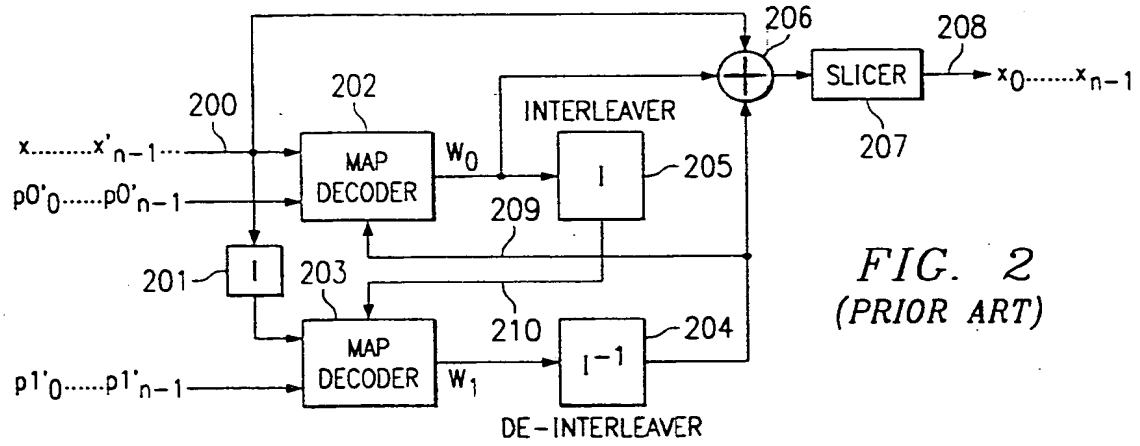

FIG. 2
(PRIOR ART)

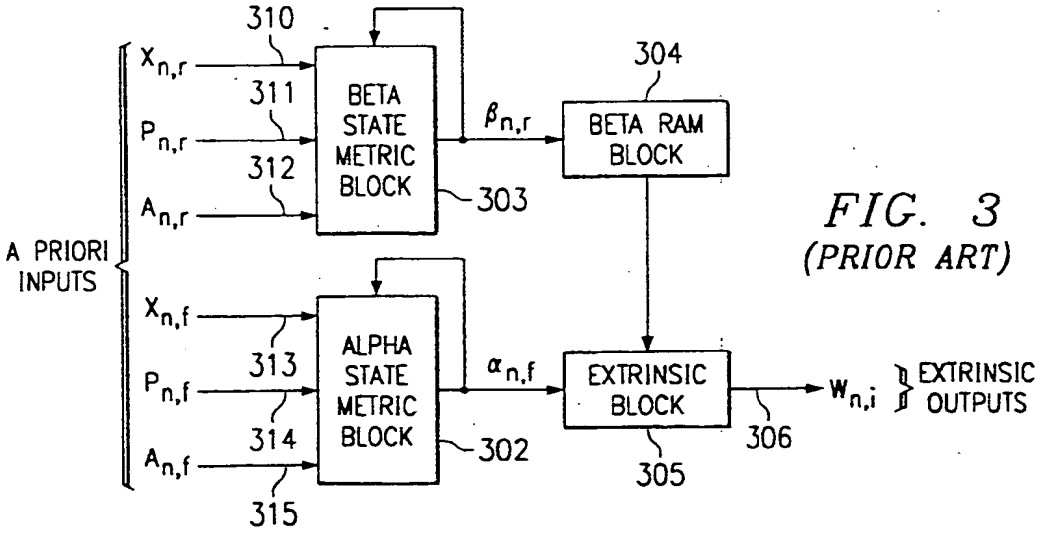

FIG. 3
(PRIOR ART)

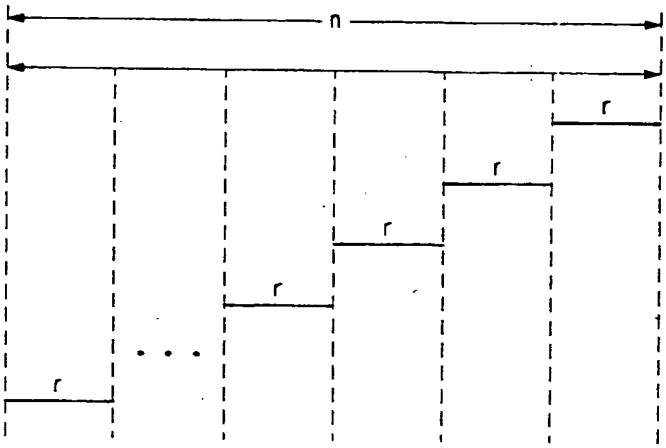

FIG. 4
(PRIOR ART)

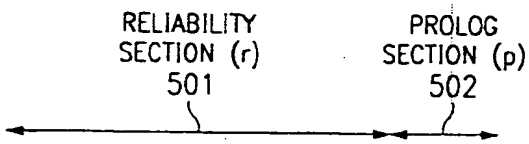

FIG. 5
(PRIOR ART)

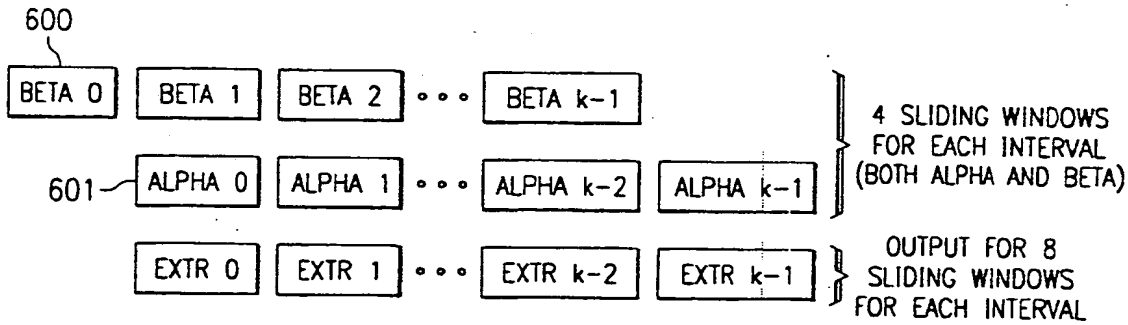

FIG. 6
(PRIOR ART)

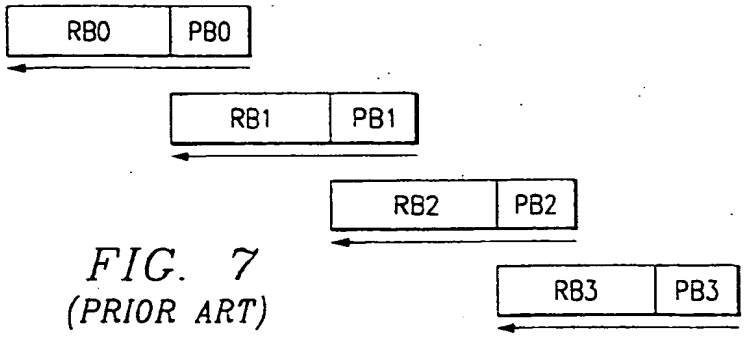

FIG. 7
(PRIOR ART)

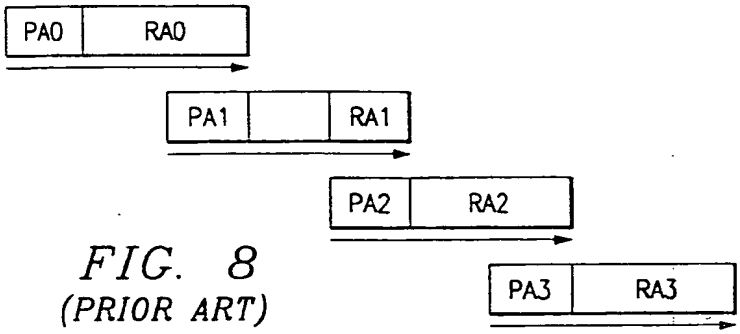

FIG. 8
(PRIOR ART)

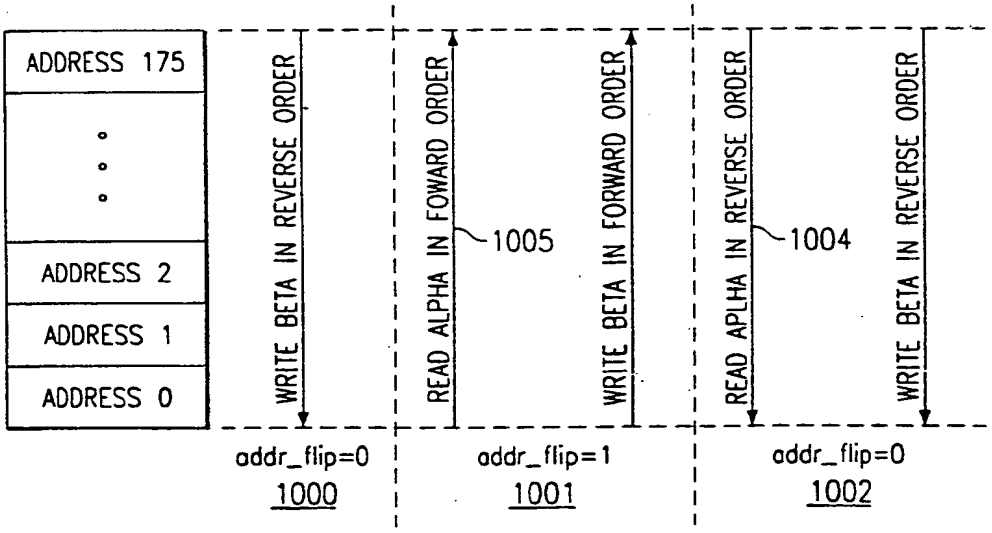

FIG. 10

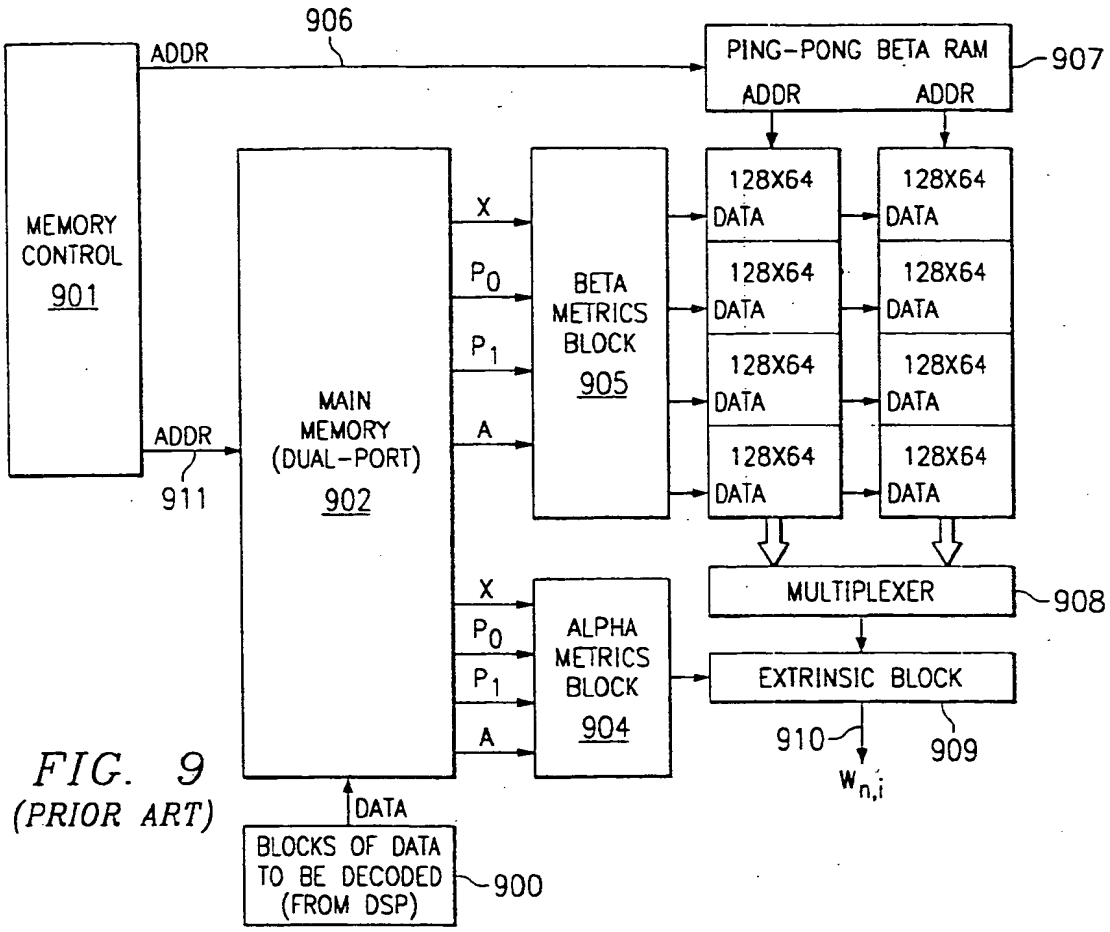

FIG. 9
(PRIOR ART)

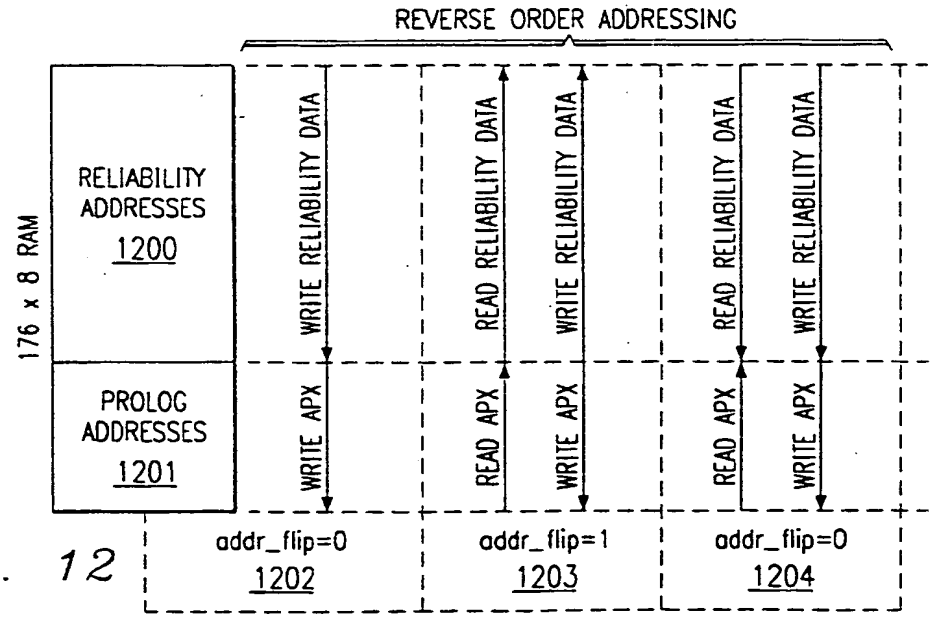

FIG. 12

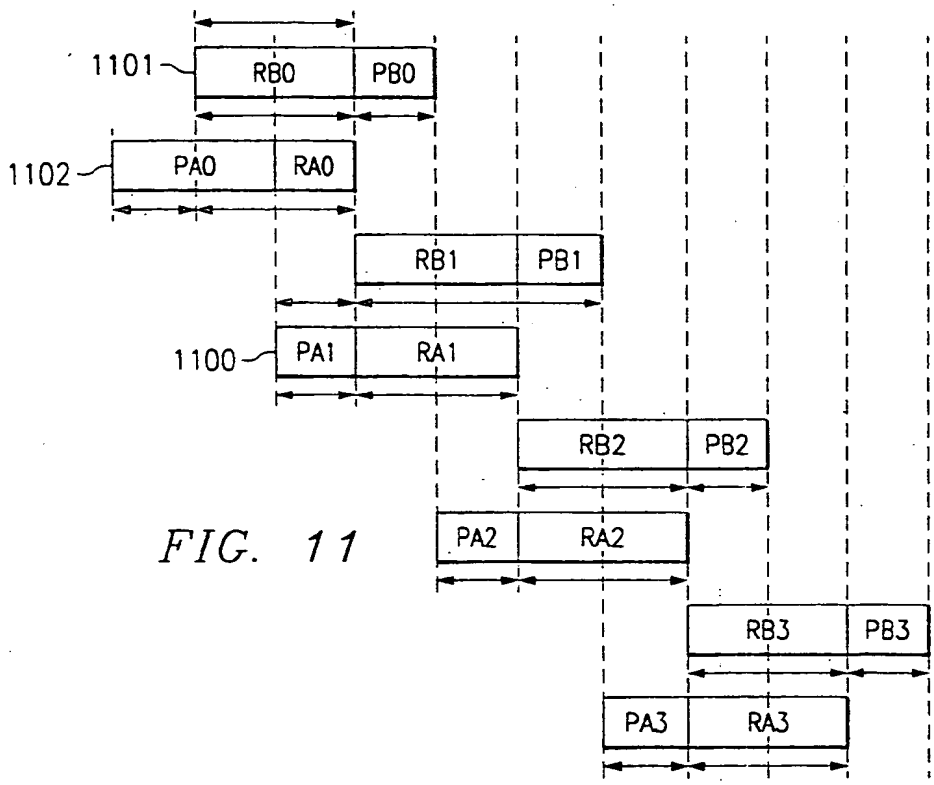
1101
RB0
PB0
1102
PA0
RA0
RB1
PB1
1100
PA1
RA1
RB2
PB2
PA2
RA2
RB3
PB3
PA3
RA3

FIG. 11

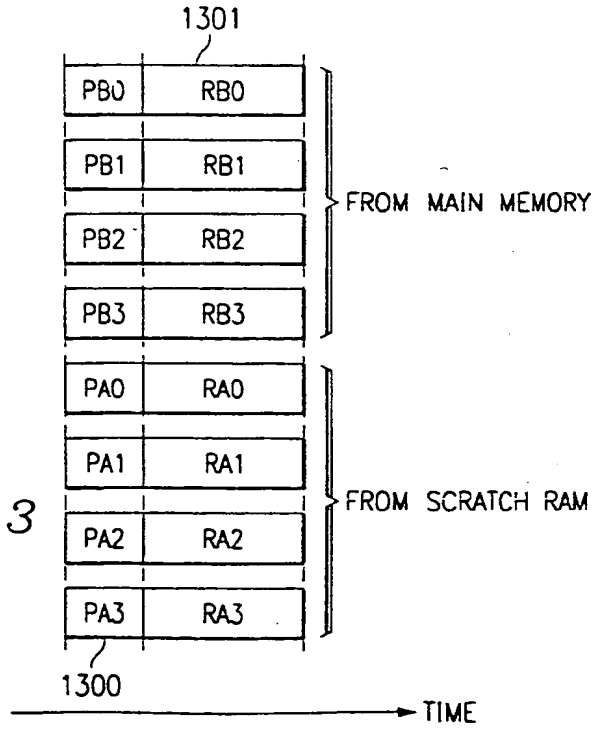
1301
PB0
RB0
PB1
RB1
PB2
RB2
PB3
RB3
FROM MAIN MEMORY
PA0
RA0
PA1
RA1
PA2
RA2
PA3
RA3
FROM SCRATCH RAM
1300
TIME

FIG. 13

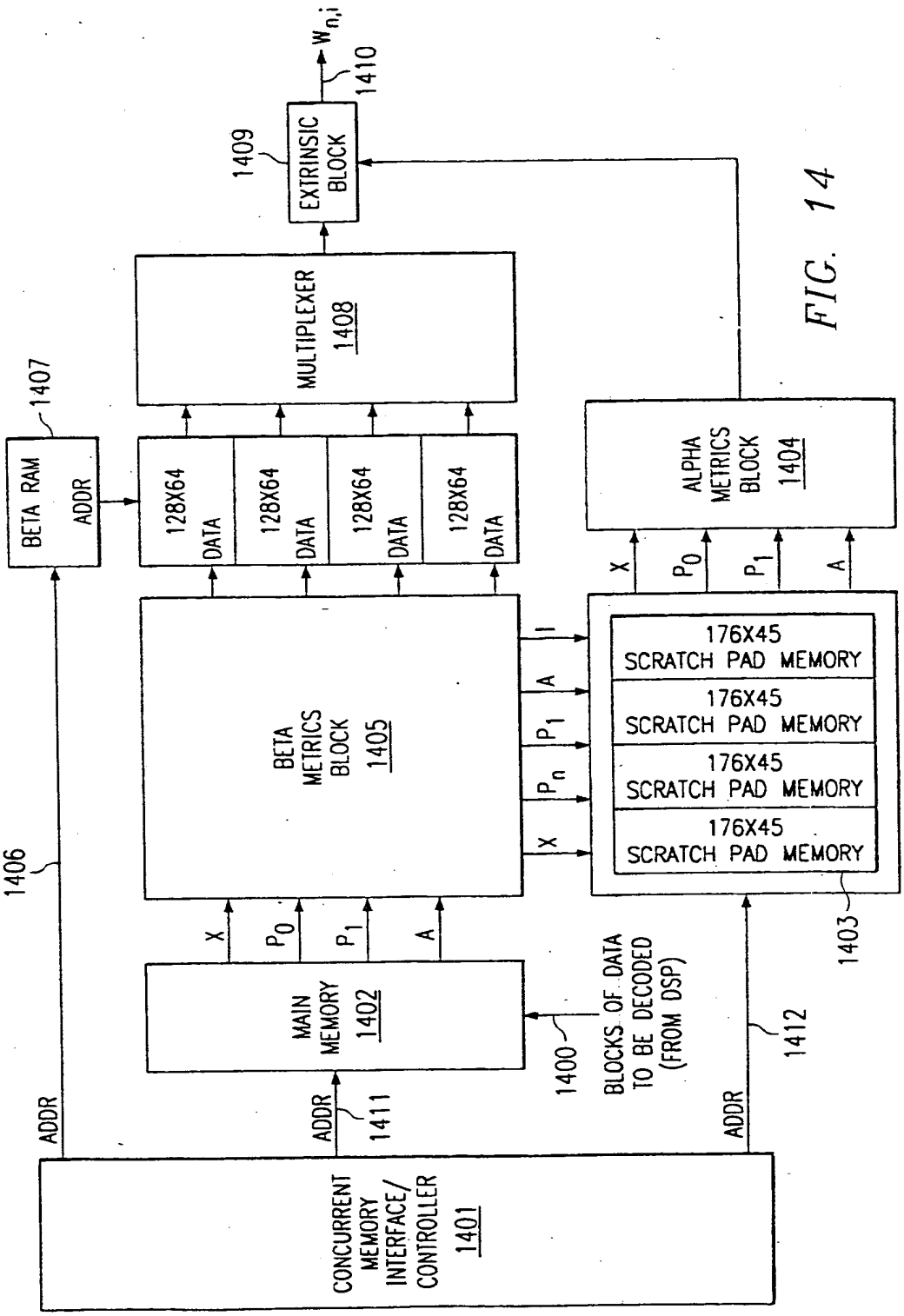
$W_{n,i}$
1410
1409
EXTRINSIC BLOCK
MULTIPLEXER 1408
1407
BETA RAM ADDR
128X64 DATA
128X64 DATA
128X64 DATA
128X64 DATA
ALPHA METRICS BLOCK 1404
X
P_0
P_1
A
BETA METRICS BLOCK 1405
I
A
P_1
P_n
X
176X45 SCRATCH PAD MEMORY
176X45 SCRATCH PAD MEMORY
176X45 SCRATCH PAD MEMORY
176X45 SCRATCH PAD MEMORY
1406
1403
MAIN MEMORY 1402
1400
BLOCKS OF DATA TO BE DECODED (FROM DSP)
1412
ADDR
1411
CONCURRENT MEMORY INTERFACE/ CONTROLLER 1401

FIG. 14